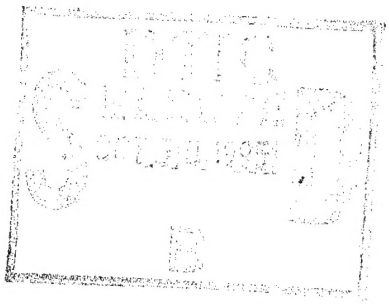
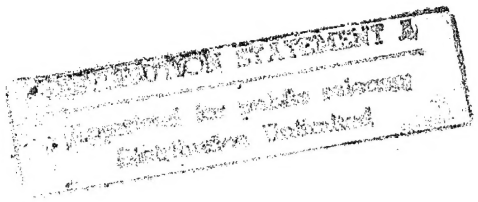


TEAM REUSE

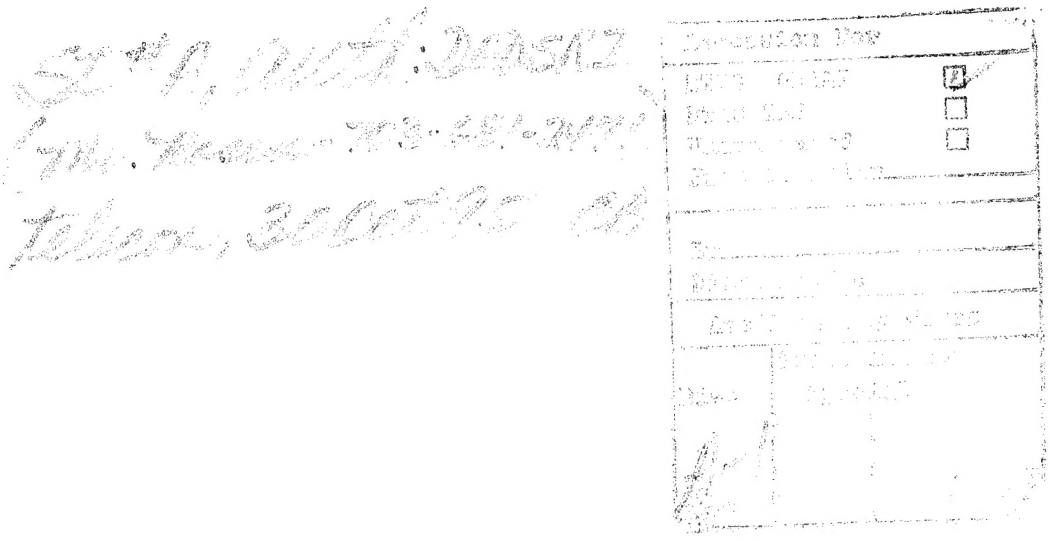
SOFTWARE REUSE EXECUTIVE PRIMER

May 17, 1995



produced by the

DOD Software Reuse Initiative
Program Management Office
5600 Columbia Pike
Falls Church, VA 22041



1st Edition, September 30, 1994
2nd Edition, December 2, 1994
3rd Edition, February 8, 1995
4th Edition, March 22, 1995
5th Edition, April 6, 1995
6th Edition, May 5, 1995
7th Edition, May 17, 1995

The information in this document is subject to change without notice.

Document No:	*PD427.6:17-MAY-95*
Contract No:	*DCA100-93-D0071*
Delivery Order:	*51*
CDRL No:	*A0005*

What is in this Primer?

The Software Reuse Executive Primer *provides a simple and understandable overview of software reuse.*

This Primer presents the issues and benefits involved in transitioning to reuse-based software acquisition, development, and maintenance processes, and answers frequently asked questions about software reuse. Technical and organizational issues are discussed along with successful reuse management techniques.

- No prior knowledge of software reuse by the reader is assumed.
- Acronyms are listed in Appendix A. The Bibliography in Appendix B lists references and other software reuse publications. Software reuse points of contact (POC) are listed in Appendix C.
- Occasional mention of commercial companies and product names in the course of providing examples, is not to be considered an endorsement of these companies or products by the Department of Defense (DOD) Software Reuse Initiative (SRI) Program Management Office (PMO).
- This Primer is not intended to replace authoritative plans, policies, or directives. DOD software reuse plans, policies and directives are found in formal documents published by the Office of the Secretary of Defense (OSD), the SRI PMO, the Services, and DOD Agencies.

Why Do You Want to Read This Primer?

Because DOD acquisition managers are responsible for implementing software reuse.

The *Software Reuse Executive Primer* was written to answer the following seventeen questions posed by DOD acquisition managers relative to software reuse in simple language:

- What is software reuse?
- Why is software reuse important?
- Where has it paid off?
- Why haven't we taken advantage of software reuse before?
- What do I reuse?
- What are the benefits of reuse?
- What are the barriers to reuse?
- What are the key non-technical issues?
- What are the key technical issues?
- What is the DOD's strategy for reuse?
- How do I know if I am ready for reuse?
- Where do I get help to kick off a reuse program?
- What are the upfront investments?
- What are common mistakes and how do I avoid them?
- How do I measure my return on investment?
- What are my next steps?
- Where do I go for more information?

This Primer is aimed at whetting the appetite of those acquisition managers responsible for implementing software reuse within both the weapons systems and information systems communities of the DOD. Managers should be interested in reuse because they are looking for ways to reduce cost, accelerate schedules, and improve software quality. Program Managers, Program Executive Officers, and Direct Reporting Program Managers fall into this category as does anyone else who may benefit by implementing a reuse program.

What is Software Reuse?

Software reuse is defined as the process of implementing or updating software systems using existing software assets. [NIST]

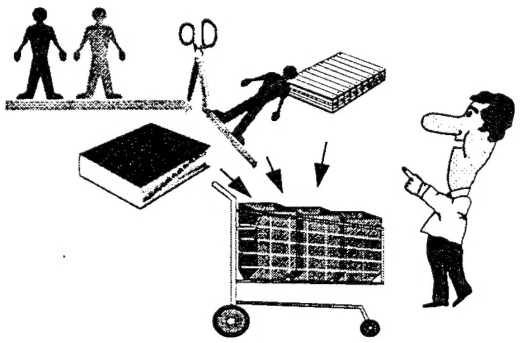
Software reuse is the application of reusable software assets to more than one software system. Software reuse may occur within a software system, across similar software systems, or in widely different software systems. [NIST]

A reusable software asset is a software product that is cataloged in a reuse library. [NIST] This includes, for example, models of distinct functional areas (i.e., domains), domain architectures, architectures, requirements, designs, code, databases, database schemas, documentation, user manuals, and test suites.

Why is Software Reuse Important?

- *Software reuse provides a basis for dramatic improvements in quality and reliability, speed of delivery, and in long-term decreased costs for software development and maintenance.*
- *Industry and DOD have demonstrated that software reuse generates significant return on investment by reducing cost, time, and effort while increasing quality, productivity, and maintainability of software systems throughout the software life-cycle.*

The Department of Defense and the Congress have stressed the importance of reducing the cost, time, and effort required to build and maintain software systems for the DOD. This applies to both in-house and contractor-developed systems. Software reuse has been demonstrated by Industry and the DOD to provide one of the greatest returns on investment (ROI) for reducing cost, time, and effort while increasing quality, productivity, and maintainability throughout the software life-cycle.



When software acquisition, development, and maintenance organizations develop reusable software products, they will do more "shopping for the best fit" when preparing to work on future software projects.

DOD software development costs have outstripped hardware costs and are continuing to grow. The major factors contributing to this growth of software costs are the continuing increase in the size and complexity of software systems and an international climate that calls for rapid adaptation to new situations. Software reuse offers significant potential to hold down future costs by taking advantage of previously developed software and by developing software designed for future reuse.

A significant challenge to both DOD management and technical staff is to improve the quality and reliability of an organization's systems in an era of decreasing budgets. The challenge is intensified by the ever increasing complexity and functionality of modern information systems. One response to this situation that has proven successful is the integration of software reuse principles into the software engineering process.

Software reuse provides a basis for dramatic improvement in the way software systems are developed and maintained over their life-cycle. Reusable software requires carefully analyzed and structured design that withstands thorough testing for functionality, reliability, and modularity. Accordingly, improvements manifest themselves in increased quality and reliability and in long-term decreased costs for software development and maintenance.

Reuse reduces the risks and costs of software acquisition, development, and maintenance.

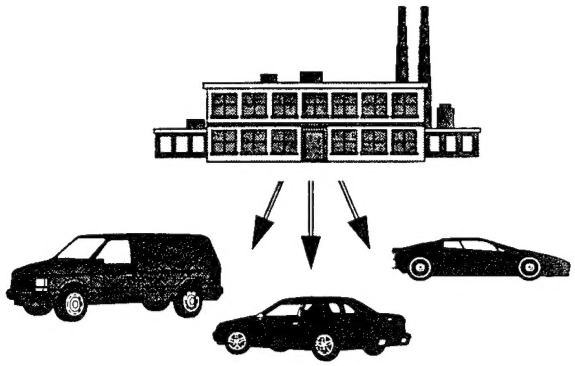
Other benefits of adopting reuse include improved interoperability and support for rapid prototyping activities. A well run reuse program decreases initial risk for development, maintenance, and acquisition activities by taking advantage of software already proven to be functional and reliable through prior usage.

Reuse is already an integral part of every engineering discipline. For example, mechanical engineers do not design a combustion engine from scratch for each car rolled off an assembly line; software engineers do not recreate the interaction between an application and its screen icon for each new product;

and aerospace engineers do not build solid rocket boosters from ground zero for each space shuttle. In all of these examples, the architecture and design of an item is reused to produce and manage a "product line."

Reusable software also can be acquired, developed, maintained, and managed via a "product-line" approach. For example, during this era of rightsizing we have to build systems quickly and cheaply. One strategy for doing this is to move to product-line techniques like commercial firms use to achieve economies of scale.

The product-line approach will increase the software acquisition, development, and maintenance community's responsiveness to DOD's plans and requirements for the future. The anticipated result is a DOD capability to deliver more responsive, higher quality products (systems) to the end user more quickly, at reduced cost, and with less risk because common software is reused.



A product-line approach similar to that of automobile manufacturing can be adapted to software acquisition, development, and maintenance processes.

Where Has It Paid Off?

Industry has realized many instances of significant payoff by instituting software reuse practices.

Hewlett Packard, since instituting software reuse in the mid-1980s, has experienced quantifiable benefits such as:

- Quality - Defects for reused code were only one-fourth the defects for new code. [IEEE]
- Productivity - Developing systems with new and reused code yielded a 57% increase in productivity over developing systems with only new code. [IEEE]
- Time-to-Market - With equivalent development effort, using reusable software assets required 42% less time to deliver the final product.

Computer Science Corporation developed the Upper Atmosphere Research Satellite (UARS) telemetry simulator using a design that allowed for future reuse. The UARS reusable assets have subsequently been reused in three other projects:

- The Extreme Ultra Violet Explorer (EUVE) telemetry simulator reused 89% of UARS code without modifications. Labor hours were reduced 67% while productivity was three times that on the original UARS development project. An order of magnitude improvement in software quality was achieved with a final error rate less than one-tenth the original UARS telemetry simulator project's error rate.
- In the Solar, Anomalous, and Magnetospheric Particle Explorer (SAMPEX), 85% of the UARS telemetry simulator architecture was reused. Staff hours on the project were 30% less than EUVE's, and the error rate was reduced similarly.
- The WIND/POLAR telemetry simulator used 55% of the reusable UARS telemetry simulator architecture. Schedule, effort, productivity, and quality estimates were improved like those of EUVE and SAMPEX.

TRW Inc.'s National Universal Network Architecture Services project had an estimated $6 million upfront investment in performance optimization, design for reuse, and portability. Using a second generation product, TRW saw a two-fold increase in productivity over typical software developments, due to a decrease in the volume of software errors found in initial test configurations and a reduction in the average time required to correct errors.

Why Haven't We Taken Advantage of Software Reuse Before?

Reusing software to reduce cost, speed development, and increase quality is not a new concept.

Software reuse has existed conceptually and practically in a variety of forms and methodologies since the early years of software development. Early software reuse practices were focused on reusing source code only. This often proved cumbersome and inefficient as storage and access technology was limited. Now, technology has progressed to the point where virtually all categories of software assets can be stored and accessed efficiently, thus making software reuse a viable software engineering discipline.

Software reuse doesn't just happen. Both management and technical issues must be tackled to transition successfully to a reuse-based culture. Once these issues are managed, the benefits discussed above can be garnered.

What Do I Reuse?

Any product from the software life-cycle can potentially be reused.

Reusable assets include, for example, domain models, domain architectures, requirements, design, code, databases, database schemas, documentation, user manuals, and test suites. Reusable software assets typically are cataloged and stored in a reuse library. The catalog entries include identification and descriptive data that helps the user to determine an asset's suitability for reuse. [V&S] By developing software systems with reuse in mind, one can take advantage of standards-based architecture methodologies that will support reuse of entire designs and whole subsystems.

What Are the Benefits of Reuse?

Lower cost systems.

Software reuse will yield significant benefits to the software acquisition, development, and maintenance communities. Commercial firms and government organizations have demonstrated that software reuse will:

- Improve the quality and reliability of software-intensive systems;
- Provide earlier identification and improved management of software technical risk;
- Shorten system development and maintenance time;
- Get products to market sooner;
- Increase productivity; and
- Reduce cost.

What are the Barriers to Reuse?

- *Non-technical and technical barriers to software reuse exist.*
- *The SRI PMO is working to remove these barriers to reach the SRI objective of institutionalizing software reuse within DOD.*

Non-technical Barriers

Non-technical barriers to reuse involve management issues, reuse standards, acquisition issues, training and education issues, and library reuse issues. These non-technical barriers continue to cause the greatest delay in DOD-wide acceptance and utilization of proven, documented software reuse concepts and methodologies. [STRAT]

The SRI PMO is overcoming non-technical barriers by providing a Software Reuse Business Model (SRBM), recently developed by the U.S. Army Space and Strategic Defense Command, that can be tailored to a variety of scenarios and sets of business practices. [STRAT]

To assist the acquisition process, the SRI PMO has sponsored the development of software reuse business models and will publish a Program Manager's Reuse Issues Handbook. The SRI PMO also encourages using incentives to accept risks associated with incorporating software reuse concepts

and procedures throughout the software development life-cycle. [STRAT]

Technical Barriers

Technical barriers to reuse involve technical processes, methods, tools, standards, and technology-based environments for software systems acquisition, development, and maintenance. Software reuse is not a stand-alone technology — it must be pervasive throughout the life-cycle. [STRAT]

The SRI PMO is overcoming technical barriers by developing a reuse-based software systems engineering paradigm and expediting the rapid transfer of reuse technology. This involves identifying best practices in Government and Industry; providing an integrated software engineering environment including the tools to support domain engineering and reuse; and providing a clearly defined technology roadmap that identifies critical technology to support software reuse. [STRAT]

What Are the Key Non-Technical Issues?

- *A reuse-focused culture is required.*

- ***Reuse management and infrastructure***. Although software reuse technology is constantly improving, we have to put in place management structures to absorb, channel, and take advantage of it. [AMPROG]

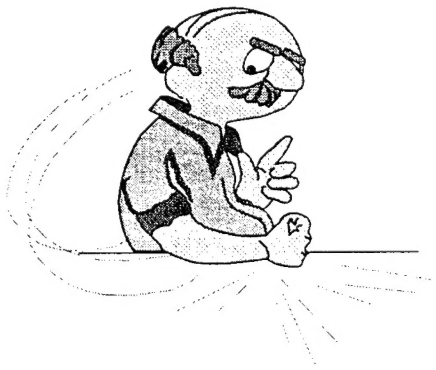
- ***Committing inadequate resources for reuse.*** Reuse doesn't just happen. Resources must be allocated to acquire, build, maintain, manage, and upgrade domain knowledge, architectures, and support tools.

- ***Assuming any reuse is good reuse.*** Directives to reuse any software available compete with, and eventually can defeat, engineering and business objectives. Reuse should be well planned, systematic, reliable, and should add value to the project. Reuse goals should be established that support achieving business and engineering objectives.

- ***Using project-by-project opportunistic ad hoc reuse only.*** Project-by-project opportunistic ad hoc reuse, although a legitimate short-term tactic, will not provide the long-term benefits of a systematic software reuse program. To reap maximum benefits, reuse must be systematic and considered across related systems (vertical domain) and across common features of possibly unrelated systems (horizontal domain). A hybrid domain (with some vertical and some horizontal aspects) promises the greatest rewards for systematic reuse.

- ***Populating architectures and repositories.*** Architectures represent candidate solutions for whole families of related systems within a domain. Populating these architectures entails design decisions that reflect the architecture. Populating repositories with high-quality reusable software as it is developed is an integral part of a long-term investment strategy and is critical to realizing high economic and technical payoff in future software efforts.

- ***Parochial attitudes.*** Software engineers still tend to mistrust assets produced by someone else. They need to be continuously motivated to reuse assets, not redevelop them. To address this issue, certification schemes have been developed and used by several organizations. [AMPROG]



Change attitude from "not invented here" to "we reuse here".

- ***Not rewarding developers for reusing software when developing <u>with</u> reusable assets, and when developing <u>for</u> future reuse.*** Contract award fees and productivity metrics should be established to encourage software engineers to develop software systems <u>with</u> reusable software, and to design components of new soft-

ware systems for future reusability. When developing for future reuse, a good rule of thumb is that reusable products will take twice the development effort as that required for developing non-reusable products. Hence, reusable products are initially more expensive, but result in downstream cost avoidance for related software components in the product line. Software reusability should be aimed at enterprise productivity across a product line of systems or families of related systems, rather than at individual or project-level productivity.

- ***Using short-sighted metrics.*** Contract award fees are commonly based on the amount of software reused. A more appropriate award fee would be based on the percentage of final delivered software that is composed of *reusable* software. This latter metric encourages contractors to populate and maintain architectures and repositories in support of future reuse efforts.

What are the Key Technical Issues?

Associated with architectures:

- *Standards for representation.*
- *Tools to assist modification and integration of multiple structural views; and automated change mechanisms.*

Associated with reusing software assets of all types (architectures, designs, documentation, code, etc.):

- *Asset production and configuration management.*
- *Asset locators.*
- *Asset control.*

Issues Associated with Architectures

The wide-spread use of architectures to guide software development is relatively new. Therefore, automated tools to assist in their development, maintenance, and modification are not commonly available. However, research in the field of enterprise integration may yield advances in techniques to integrate models representing varying structural views.

More than likely, tools will precede standards and organizations responsible for maintaining domain knowledge. Such tools will automatically ripple changes throughout the life-cycle product line (e.g., architectures, domain models, data models, design documentation), and while they will not be available for some time, they will allow organizations attempting to develop and maintain architectures to have access to some technological support.

The SRI PMO supports the concept of developing system architectures that will provide a framework for the design of reusable software. Plans call for identification and acquisition within two years of "best of breed" architectures for the initial, high-leverage product lines. After five years, the objective is to have at least one complete, validated architecture for each active product line. [STRAT]

Issues Associated with Reusing Software Assets of All Types

System development and maintenance organizations have similar technical challenges: finding, understanding, modifying, integrating, and composing assets; developing, populating, maintaining, and managing architectures; and creating and managing software components such as designs and documentation.

Automated tools should be selected to provide the greatest capabilities in location, selection, use, and control of the assets. Promising technologies should be identified and tools procured that give the most functionality. These tools speed development and help bring about a complete asset inventory with standardized style and structure. This facilitates maintainability and, most of all, use of the assets.

Organizations need to develop and maintain configuration management information (e.g., an automated index of all architectures and other components). Brief, precise, and functional descriptions should be included. This will assist in reusing these assets and will identify opportunities for future reuse.

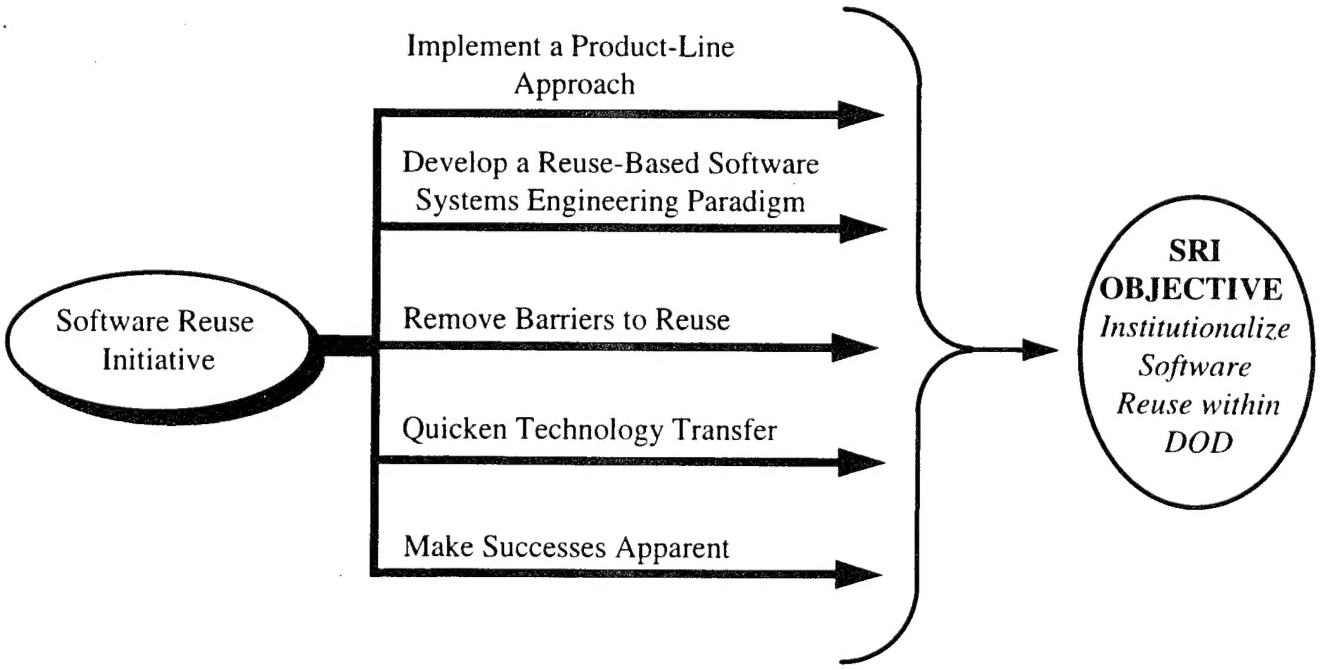
The SRI PMO has established the Defense Software Repository System (DSRS) for use by all DOD components. This repository will provide on-line support to organizations desiring to use proven and reliable software assets.

What is the DOD's Strategy for Reuse?

- *The DOD SRI has established an incremental strategy for institutionalizing software reuse within the DOD.*
- *The DOD SRI will use a five-pronged approach to implement the strategy.*

The DOD SRI's incremental strategy for adopting software reuse will produce initial successes within two years. By the end of five years, the strategy is expected to provide the foundation to achieve widespread acceptance of software reuse as an integral component of software systems acquisition, development, and maintenance. The SRI will use the five-pronged approach illustrated in the accompanying figure. The five thrusts are based upon lessons learned to date, including current plans and programs, user requirements, and Congressional guidance. An overview of each thrust follows: [STRAT]

- ***Implement a Product-Line Approach*** will reconstruct the way DOD organizations acquire, develop, and maintain software systems. A product line is a set of software systems with common architectures that satisfy the mission requirements of one or more domains. A product-line approach involves developing and applying reusable assets among software systems possessing similar architectures. [STRAT]
- ***Develop a Reuse-Based Software Systems Engineering Paradigm*** will build and maintain a technical foundation for reuse-based software systems engineering. This will be accomplished by identifying the best practices and supporting environments for conducting software reuse. This thrust will institutionalize reuse concepts by making reuse an integral part of software systems engineering. [STRAT]
- ***Remove Barriers to Reuse*** will identify and remove non-technical barriers such as legal, contractual, organizational, and economic, that inhibit the product-line approach to software acquisition, development, and maintenance. Incentives to speed reuse adoption also will be provided. [STRAT]
- ***Quicken Technology Transfer*** will develop processes, products, service solutions, and appropriate partnerships for expediting the adoption and institutionalization of software reuse. Quick technology transfer also will be stimulated through a variety of partnerships among the DOD, other Government organizations, Industry, and Academia. [STRAT]



Five-Pronged Approach to Institutionalize Software Reuse within DOD

- ***Make Successes Apparent*** will publicize software reuse successes and provide sufficient information to promote reuse awareness within the DOD. Widespread dissemination of reuse information will be provided to further the understanding of reuse and its benefits. [STRAT]

How Do I Know if I am Ready for Reuse?

- *Select projects within a distinct domain (functional area) to minimize risk.*
- *Domain-specific reuse allows for an iterative, incremental adoption of reuse. This evolutionary method has less risk associated with it than a revolutionary, "big bang" method.*

You are ready for reuse when the associated risks are at an acceptable level. Analysis of the domain in which you are operating is the key to determining that level.

Selecting projects within a single domain leads to achieving systematic reuse. Limiting the focus of the software engineering effort to a single well defined and well understood set of functions, increases the opportunities for successful software reuse. Thus, reusable software assets can be identified precisely and utilized efficiently. The narrowness of the domain reduces the investment in the reuse repository because the domain is well understood, the problem is well understood, and engineers can "get their minds around it". If a domain is too large to be managed, then it should be sectioned into subdomains.

The type of domain also is important. A low-risk domain for reuse is one in which the technology and the products are relatively stable or evolving at a manageable pace. A high-risk domain for reuse is one in which the underlying technology is rapidly changing (e.g., system software for personal computers and workstations).

Selecting a single domain project allows an organization to refine its reuse techniques. Experience is developed within a single domain. Metrics can be uniform throughout the domain's projects and are easily compared. Examples of using metrics are:

- For a software acquisition organization, the proposal and vendor evaluation criteria will have been tested and recalibrated, and their usefulness and veracity will have been demonstrated repeatedly.
- For a software development organization, as more software assets are made available for reuse in the repository, software development costs will begin to decrease.
- For a software maintenance organization, design rationales and implementation decisions for the systems within the domain can be evaluated and compared in order to justify accepting or rejecting modification requests.

Where Do I Get Help to Kick Off a Reuse Program?

- *Call your Service/Agency software reuse POC.*
- *Coordinate your software reuse projects among different functional areas (i.e., domains).*
- *Participate in your Service/Agency reuse planning.*

Get help by calling your Service/Agency software reuse POC. They are listed in Appendix C and are available to discuss starting a reuse program.

It is important to coordinate your software reuse projects among different functional areas. This enables different program managers to take advantage of ongoing reuse development efforts. The Services'/Agencies' reuse plans will establish an initial framework from which all reuse efforts within the Service/Agency will be coordinated. Priorities will be defined and will influence budgets. Accordingly, it is essential for acquisition managers to participate in this planning process.

What are the Upfront Investments?

Successful software reuse requires an upfront investment.

Opportunities to implement software reuse must be seized as early as possible. The specifics of the upfront investment opportunity vary among acquisition, development, and maintenance managers.

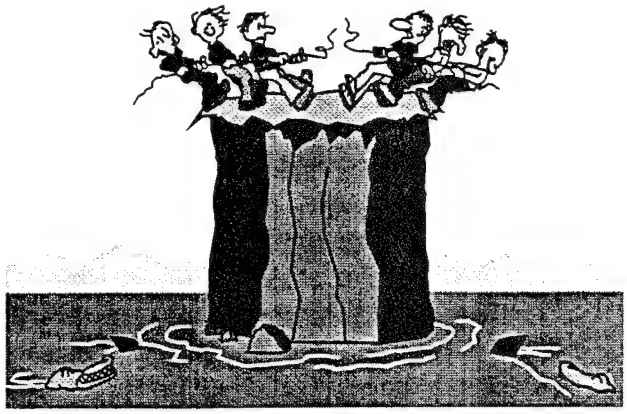
For acquisition organizations, the upfront investment will be in redesigning the current evaluation system for proposals and vendors so that bidders will be rewarded for applying reuse. Criteria should be included in the solicitation that will: (1) forecast the percentage of reusability expected for a project; and (2) accurately capture the breadth and depth of the bidder's experience with reuse. During the proposal evaluation phase, acquisition organizations should ensure that the bidder's software reuse plan is executable and capable of providing desired levels of software reuse performance. Some organizations have already pioneered these changes so your upfront investment can be reduced by copying or adapting from these efforts.

For development and maintenance organizations, the additional upfront investment will be the cost of domain engineering (including architecture development), tool acquisition, and integration. Also, different testing and quality assurance processes often are needed when reusing software. After contract award, development and maintenance organizations should establish monitoring procedures that ensure the contractor is meeting the software reuse performance standards described in the statement of work.

High-quality repositories may be another upfront investment for development and maintenance organizations. However, existing repositories of reusable software, such as the DSRS operated by the SRI PMO, are available for use by DOD components.

For organizations opting to develop and maintain their own repository for reusable software, automated browsing tools should be acquired or developed to facilitate search and retrieval. Also, configuration management tools should be incorporated into repositories in order to trace an asset to the systems in which it was used.

There is great pressure for early high-payoff success stories, but the real measure of success considers the entire life-cycle. Reuse benefits are real and are documented in earlier sections of this Primer. Not only will your program benefit from software reuse, but the benefits multiply as follow-on programs take advantage of your work. To ensure a high payoff from software reuse:



The pressures for early high-payoff and long-term success can divide and conquer unwary organizations.

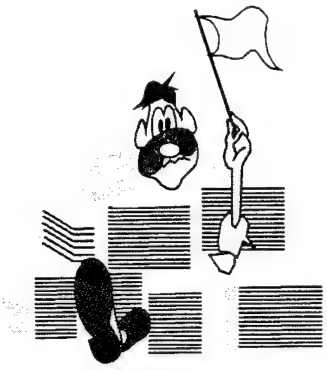
- Invest in a long-term systematic software reuse strategy throughout the software development cycle;
- Exploit high-payoff ad hoc software reuse opportunities as they occur; and
- Be realistic; do not promise too much too soon.

What are Common Mistakes and How Do I Avoid Them?

Plan and create an infrastructure to avoid mistakes.

- ***Allowing inadequate configuration management of reuse assets; which assets are used in which systems?*** Configuration management assists in making the decision to use a reusable asset and in developing data for estimating an organization's return on investment. Establish appropriate configuration management mechanisms.

- ***Inadequately controlling what is put in the reuse library.*** Issues involved here are quality of assets, breadth of functionality, limiting repetition, and proper asset documentation. Establish appropriate configuration management mechanisms over the library.
- ***Using inadequate search/browse/lookup mechanisms in the reuse library.*** If you can't find it, you can't use it. Invest in support tools.



Plan, create, and manage an adequate infrastructure of support tools to avoid common technical mistakes.

- ***Incorporating software assets that are too large and/or have too many unanticipated side effects.*** Smaller software assets are better for reuse. They have fewer unanticipated side effects and are easier to categorize. In addition, they require less overhead to maintain.
- ***Using reusable software that requires excessive overhead to compile, link, and execute.*** If it is difficult to integrate an asset into the project, it won't be used.
- ***Bypassing domain engineering.*** Neither top-down nor bottom-up design is adequate to capture the benefits of reuse. Domain engineering must be accomplished. It is often referred to as middle-out design.
- ***Expecting a domain to satisfy all of the requirements all of the time.*** If the scope of a domain is too broad, then all requirements cannot be captured and satisfied adequately for all systems within the domain. Hence, systems developed from too broad of a domain definition will find that requirements, such as performance, will most likely suffer.
- ***Allowing no flexibility for exceptions and overrides.*** "All or nothing" reuse does not provide any flexibility to tailor reusable software to satisfy a new requirement. For software code, as an example, the result is substantial amounts of "dead code" (i.e., unused source code that is compiled and bound into the executable form). This creates unnecessary overhead and deters reuse.
- ***Allowing undocumented interfaces.*** This results in wasted time and money as software engineers must break into the asset's "black box" to determine its impact on the project's design or its execution environment.

How Do I Measure My Return on Investment?

- *The benefits that will best justify investing in a reuse program may be organization-unique (e.g., cost, productivity, quality, and reliability).*
- *Metrics, algorithms, and processes should be developed to measure return on investment (ROI).*
- *Historical and current data must be collected to measure ROI.*

Baseline measurements (e.g., cost, productivity, reliability, quality) must be defined and the values assigned for historical and current projects. These metrics must be chosen carefully and the algorithms used to produce them should be examined to determine if they are able to truly represent the value of reuse. The metrics selected to measure the benefits of transitioning to reuse should address both process (e.g., the depth and breadth of the product-line lifecycle) and product (e.g., the number of trouble reports and engineering change proposals).

The measurements that will best justify investment in the reuse program may be organization-unique. For example, reliability measures, such as mean time between failures, mean time to repair a defect, and defect backlog, would be especially appropriate for a software maintenance organization

responsible for a weapon system. On the other hand, a software development organization would probably be better served by using cost and productivity measures.

Metrics should be functional enough to be used as gauges for project success. They must be economical to collect and report, and collection should be automated. Mechanisms should be established to accumulate an organizational database of historical financial data relative to software production and maintenance, including reuse activities.

The internalization of reuse principles into an organization's methods will evolve over time. Thus, the ROI seen at 12 months will be less than the ROI seen at 24 or 36 months. Often, the ROI is accrued on subsequent projects.

What are My Next Steps?

- *Contact your Service/Agency software reuse POC.*
- *Develop a reuse plan.*
- *Use the SRI PMO resources.*
- *Provide leadership and resources.*
- *Reward reuse, not alternatives.*
- *Reorganize to facilitate reuse.*

The next logical step is to contact your Service/Agency software reuse POC (Appendix C), determine if your organization is ready for the transition, and develop a transition plan using the strategies discussed in this Primer. The SRI PMO also can provide assistance and resources towards developing a reuse plan.

A summary of management strategies to implement and sustain software reuse within an organization follows:

- Provide proactive leadership and resources at the beginning of system development projects to encourage reuse.
- Establish reuse goals that support achievement of business and engineering objectives.
- Provide a reward mechanism instilling a desire to reuse software for development, maintenance, and acquisition activities.
- Plan for long-term investment in reuse.
- Treat technology transition as a project: manage it, resource it, schedule it, and measure progress.
- Adopt a product-line approach to software acquisition, development, and maintenance.
- Establish a systematic reuse program.
- Plan, implement, and analyze incremental efforts to adopt reuse.
- Establish cost models and program metrics to facilitate decisions on reuse.
- Provide comprehensive and top-quality resources to development programmers.
- Establish organizational standards for design, reuse, and programming. Validate them.
- Enforce validated standards over whole development organizations.
- Exploit ad hoc reuse opportunities while concurrently building the infrastructure necessary to take advantage of systematic reuse.
- Seek mechanisms for sharing reuse experiences and ideas (e.g., workshops, conferences, and working groups).

Where Do I Go For More Information?

- *This Primer is just the starting point.*
- *Contact the SRI PMO and Reuse Information Clearinghouse (ReuseIC).*

This *Software Reuse Executive Primer* has only scratched the surface. Further study of the topics in this Primer is recommended. The Bibliography is an excellent starting point. The SRI PMO and ReuseIC can provide information on upcoming conferences and educational opportunities. In addition, points of contact for reuse working groups can be obtained from these sources. Products developed by these groups can provide invaluable help for organizations adopting a software reuse program.

The ReuseIC is the primary distribution mechanism for reuse information operated by the SRI PMO. The ReuseIC provides public access to reuse information through Internet, a telephone hotline, and a quarterly newsletter. Information about ReuseIC access can be obtained by contacting Ms. Susan Carlson at E-mail: reuseic@sw-eng.falls-church.va.us, Fax: (703) 681-2869, and Voice: (703) 681-2464.

Appendix A
Acronyms

ACM	Association for Computing Machinery
CARDS	Comprehensive Approach to Reusable Defense Software (formerly Central Archive for Reusable Defense Software)
DISA	Defense Information System Agency
DOD	Department of Defense
DSRS	Defense Software Repository System
EUVE	Extreme UltraViolet Explorer
GAO	Government Accounting Office
IEEE	Institute of Electrical and Electronics Engineers, Inc.
NIST	National Institute of Standards and Technology
OSD	Office of the Secretary of Defense
PMO	Program Management Office
POC	Point of Contact
RAAT	Reuse Acquisition Action Team
ReuseIC	Reuse Information Clearinghouse
ReuseWG	Reuse Working Group
ROI	Return on Investment
SAMPEX	Solar, Anomalous, and Magnetospheric Particle Explorer
SIG	Special Interest Group
SPC	Software Productivity Center
SRBM	Software Reuse Business Model
SRI	Software Reuse Initiative
STARS	Software Technology for Adaptable, Reliable Systems
UARS	Upper Atmosphere Research Satellite

Appendix B
Bibliography

1. [AM PROG] "Software Reuse: The Next Silver Bullet?", D.J. Reifer, Vol 6, No. 8, *American Programmer Magazine*, August 1993.

2. Association for Computing Machinery (ACM) Special Interest Group on Ada (SIGAda) Reuse Working Group (ReuseWG) Reuse Acquisition Action Team (RAAT), "Software Reuse Success Stories", Presentation, September 3, 1993.

3. Association for Computing Machinery (ACM) Special Interest Group on Ada (SIGAda) Reuse Working Group (ReuseWG) Reuse Acquisition Action Team (RAAT), "Software Reuse Acquisition Scenarios: A Discussion Paper", July 28, 1994.

4. *Department of the Navy Software Reuse Implementation Plan* (Draft), Naval Information Systems Management Center, March 1994.

5. DISA, Center for Software, *Software Reuse Business Model (SRBM) Technical Report*, prepared by U.S.Army Space & Strategic Defense Command, Software Engineering Division, January 31, 1995.

6. DOD, *Management of the Software for Adaptable, Reliable Systems Program*, Defense Inspector General Audit 91-050, February 1991.

7. [V&S] DOD, Software Reuse Initiative, *DOD Software Reuse Vision and Strategy*, July 15, 1992.

8. DOD Software Reuse Initiative, *Technology Roadmap Volume 1: Technology Assessment*, Version 2.1, Working Draft, Defense Information Systems Agency, Falls Church, VA., January 6, 1995.

9. DOD, "Report to the Congress: US Department of Defense, DOD Software Reuse Initiatives", March 1994.

10. [STRAT] *DOD Software Reuse Initiative Strategic Plan* (Coordination Draft), 24 March 1995.

11. Government Accounting Office, "Report to the Committee on Appropriations, House of Representatives: Issues Facing Software Reuse", GAO/IMTEC-93-16, January 1993.

12. [IEEE] Institute of Electrical and Electronics Engineers, Inc., IEEE Computer Society, *IEEE Software*, "Systematic Reuse", Los Alamitos, CA., September 1994.

13. [NIST] National Institute of Standards and Technology, *Glossary of Software Reuse Terms*, NIST Special Publication 500-222, Computer Systems Laboratory, Gaithersburg, MD, December 1994.

14. Software Productivity Consortium, *Domain Engineering Guidebook*, SPC-92019-CMC, Herndon, VA, December 1992.

15. Software Productivity Consortium, *Reuse Adoption Guidebook*, Version 02.00.05, Herndon, VA, November 1993.

16. Software Productivity Consortium, *Reuse-Driven Software Process Guidebook*, Herndon, VA, November 1993.

17. Software Technology for Adaptable, Reliable Systems (STARS), *The Reuse-Oriented Software Evolution (ROSE) Process Model*, Version 0.5, Unisys STARS Technical Report STARS-UC-05155/001/00, STARS Technology Center, Arlington, VA, July 1993.

18. Taylor, David A., Ph.D., Addison-Wesley Publishing Company, *Object-Oriented Technology: A Manager's Guide*, 1990.

19. U.S. Air Force Electronic Systems Center, *Acquisition Handbook - Final Central Archive for Reusable Defense Software (CARDS)*, October 1992.

20. *U.S. Army Software Reuse Policy* (Draft), March 1, 1995.

21. U.S. House of Representatives, "Report of the House Appropriations Committee on the Department of Defense FY1994 Appropriations Bill", 103rd Congress 1st Session; Report 103-254, September 22, 1993.

Appendix C
Software Reuse Points of Contact

AIR FORCE
LtCol Michael Pait
HQ USAF/SCTT
1250 Air Force Pentagon
Washington. DC 20330-1250

ARMY
Mr. Donald W. Routten
HQ DA. ODISC4
SAIS-ADW Room 1C676
107 Army Pentagon
Washington, DC 20310-0107

LTC Gene Glasser
Army Reuse Center (ASQB-IRC)
US Army Information Systems
Software Center
Ft Belvoir, VA 22060-5456

ARPA
Mr. John Foreman
STARS/ARPA
801 N Randolph Street
Suite 400
Arlington, VA 22203

BMDO
Maj. Rick Pollard
Ballistic Missile Defense Org
7100 Defense Pentagon
Room 1E147
Washington, DC 20301-7100

DCA
Mrs. Bennie Wooten
Defense Commissary Agency
ATTN: IMP
Fort Lee, VA 23801-6300

DCAA
Mr. James White
Defense Contract Audit Agency
4075 Park Ave
Memphis, TN 38111

DFAS
Mr. Chandler Sirmons
Defense Finance and
Accounting Service
1931 Jefferson Davis Hwy
Bldg. CM3
Arlington, VA 22240

DIA
Mr. Timothy West
Defense Intelligence Agency
Bolling AFB
SY-1A Bldg 6000
Washington, DC 20340

DIS
Mr. Thomas Caine
Defense Investigative Service
2200 Deman Street
Baltimore. MD 21224

DISA
Ms. Donna Leigh
DISA/CFSW/TXE
5600 Columbia Pike
Rm 631
Falls Church, VA 22041

DLA
Ms. Lyn Kennedy
Defense Logistics Agency
DLA-CANMC
3A625 Cameron Station
Alexandria, VA 22304-6100

DMA
Ms.Pamela Krause
HQ DMA (TIM)
8613 Lee Hwy. M.S. A-10
Fairfax, VA 22031-2137

DNA
Mr. Robert G. Thomas
Defense Nuclear Agency
6801 Telegraph Road
Alexandria, VA 22310-3398

MARINE CORPS
Maj. Jerry Depasquale
Marine Corps Architecture
& Standards Division
3255 Meyers Avenue
Quantico, VA 22134-5048

NAVY
Ms. Antoinette Stuart
1225 Jefferson Davis Hwy
Crystal Gateway 2
Suite 1500
Arlington, VA 22202-4311

NSA
Mr. Jay Ferguson
Code J3
NSA/CSS
National Security Agency
9800 Savage Road
Fort Meade, MD 20755

OGC/DLSA
Mr. Roy Carryer
Office of Gov. Counsel
Legal Services Agency
The Pentagon, Room 3D929
Washington, DC 20301-1600

OSIA
Ms. Bonnie Smith
On Site Inspection Agency
300 W Service Road
Dulles International Airport
Washington, DC 20041

SRI
Mr. Don Reifer
Center for Software (CFSW)
5600 Columbia Pike
Falls Church, VA 22041

USUHS
Mr. Michael Karas
Uniformed Services University
of the Health Sciences
4301 Jones Bridge Road
Bethesda, MD 20814